



Between Form and Information: Early Philosophies of Computer-Aided Design

Daniel Cardoso Llach¹ 

Accepted: 9 September 2021 / Published online: 6 October 2021
© Kim Williams Books, Turin 2021

Abstract

This article draws from primary historical sources to examine the origin of, and tensions between, two postwar era modeling techniques that shaped the early history of computer-aided design: the *plex*, developed by Douglas T. Ross, and the *Coons patch*, developed by Steven A. Coons. The article shows how each of these two techniques can be seen as emblematic of a fundamentally different understanding of design—one centered on *information*, and another on *form*—crucially foreshadowing present-day practices of computational design in architecture and other creative fields. Using them to illuminate the roots of ongoing technical and cultural trajectories of architectural modeling, and their frictions, the article shows how computer modeling techniques perform not just as tools but also as discursive devices, artifacts of persuasion, and instruments for disciplinary and intellectual demarcations.

Keywords Computer-aided design · Geometric modeling · Steven A. Coons · Douglas T. Ross · Design · MIT · Design theory · Engineering · Architecture

Introduction

Computation offers what we might term a procedural ontology to a range of disciplines from mathematics, where it originated, to other scientific and artistic fields, prominently architecture, where it has been used ambivalently both in

✉ Daniel Cardoso Llach
dcardoso@cmu.edu

¹ Carnegie Mellon University, 5000 Forbes Ave. Margaret Morrison Carnegie Hall, Of. 412, Pittsburgh, PA 1521, USA

reference to concrete technologies and as an epistemic template channeling generative, descriptive, or analytical aspirations. Tracing early disclosures of this productive ambivalence,¹ this paper will reflect on two artifacts from the history of computer-aided design (CAD). The first is the “plex,” which was developed at the Servomechanisms Laboratory at MIT by mathematician Douglas T. Ross in the 1950s to help steer early software and numerical control projects, and foreshadowed object orientation (OOP) (Cardoso Llach 2015, 189). Ross described a plex as “an interweaved [sic] combination of parts in a structure... [with the purpose of representing a] thing, be it concrete or abstract, physical or conceptual” (Ross 1968: 14). The second is the “Coons patch,” an early mathematical technique for calculating curved surfaces from a set of curves, created by Steven A. Coons, which demonstrated the formal possibilities of linking computers and numerically controlled machinery, and was key to attracting a generation of researchers on both sides of the Atlantic to problems of computational geometry and design.² I will show how these two techniques can be seen as emblematic of fundamentally different understandings of design, which foreshadow the contemporary landscape of architectural and design modeling technologies. One, represented by the plex, initiates a tradition that focuses on taking advantage of the structured nature of digital information to represent artifacts or systems as a series of interlinked information attributes. Another, represented by the Coons patch, takes advantage of the same structures to enable new kinds of geometric plasticity, and new ways of computationally linking design and manufacturing.

Challenging abstract/concrete binaries, I will further discuss the realization of these techniques through a plurality of discursive and material practices involving diagrams, numerically-produced objects, texts, and (crucially) software, suggesting their ambivalence as both operative artifacts and symbolic notations is a distinctive feature of computational methods and objects. As we shall see, this ambivalence has a very tractable presence and repercussions in contemporary discourses and practices of design and computation.

The Plex

The plex was developed at the MIT Servomechanisms Laboratory by a young mathematician named Douglas T. Ross. Its story begins in 1956, with the development of Automated Programming Tool (APT). APT was a problem-oriented language intended to streamline the process of preparing the instructions for a computer to automatically operate a machine tool. Since the laboratory’s successful demonstration of a numerically-controlled milling machine in 1952, lab members including John H. Runyon and Arnold Siegl worked on this problem. They created short pieces of reusable computer code, called subroutines, which encapsulated the

¹ My use of “ambivalence” here relates to what sociologist John Law has described as “fractionality;” see (Law 2002).

² For an extended discussion of the MIT CAD Project see (Cardoso Llach 2015).

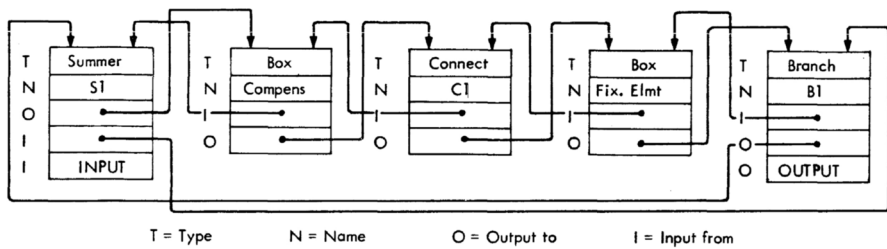


Fig. 1 An example of a modeling plex representing a control loop, from Ross and Rodriguez (1963: Fig. 12)

commands for producing specific profiles, and dealt with some of the cumbersome calculations involved in directing a three-axis milling machine using punched paper tape. But under Ross's leadership, the APT project aspired to go beyond encoding spindle speeds, tolerances, and the geometric attributes of parts. Instead, it aspired to become a scalable system capable of describing a much wider range of artifacts at a higher level of abstraction (Fig. 1). For Ross—a mathematician fundamentally concerned with representation—this meant shifting away from the design of subroutines into designing a more general language capable of representing not only points, lines, and mechanical motions, but also complex artifacts such as houses, circuits, languages, and even problems, in computable terms. The plex is a synthesis of this ambition.

A plex, the theory went, comprised three key elements: *data*, *structure*, and *algorithm*. The *data* are “units or indivisible entities in terms of which the ‘thing’s’ properties are described or measured;” *structure* refers to the relationships between the data; and the *algorithm* is “the capstone that allows the data in the structure to be interpreted, manipulated, and filled with *meaning*” (Ross 1968: 14). The algorithm relates to the behavior and the interpretation of the whole: a sort of logical rule set for operation and assembly. The plex thus codifies Ross's general theory of representation. A plex representing a line, for example, had to contain sub-entities for its starting and ending points; each point sub-entity would in turn contain values for its *x* and *y* coordinates; another sub-entity would describe the line itself as an independent element with pointers to the other two sub-entities; and so on (Fig. 1). “From the computer applications point of view,” Ross wrote in his distinctive analytical voice, “the primary problem is not how to solve problems, but how to state them” (Ross 1960: iii). Recounting the history of the concept, Ross and John E. Ward wrote that “although initial probings [sic] in these directions were much earlier, in the summer of 1960 we coined the word *plex* to serve as a generic term for these philosophical ruminations ... Our understanding of the concept of the plex has, however, been greatly enriched over the years as we have used it actively in our work” (Ross and Ward 1967: 14).

From here, searching for the plex's concrete motivations and effects, we might consider how it performs variously as a technical, institutional, and disciplinary device. First, as a discursive artifact inscribed within the intellectual and institutional spaces of cold-war science, the plex served the purpose of endowing the

worlds of materials and tools into the “clean” worlds of digital abstractions. And yet, an obstinate problem remains: the labors involved in developing and manipulating the abstract scaffoldings—the concepts, the constructs that help drive technological projects such as APT—are often themselves obscured, as are too obscured their very material substrates. A discussion of this, however, is beyond the scope of this present paper.

Finally, the plex construct helped establish a theoretical commitment among members of this group of early CAD researchers to the notion that computational descriptions bear structural resemblances to the artifacts they depict. Technically important is the fact that the hierarchical design of the plex abstraction, and in particular its dissociation between data, structure, and algorithm, foreshadowed object-oriented programming—a computer programming paradigm in which functionality is organized around abstract entities or objects, instead of procedurally along (run) time. Before the emergence of object-oriented programming, all computer programs were strictly procedural. They comprised sequences of instructions to be orderly executed by a processor. By contrast, object-oriented programs allow for the definition of “objects” with characteristics and behaviors of their own. As imagined by Ross, the plex hints at this paradigm. And yet, APT was not object- but problem-oriented, and thus it did not live up to Ross’s ambitions for the plex. True to its roots in manufacturing, it was a language of commands designed for orderly mechanical execution.

Plexes appear more prominently in Ross’s concurrent effort to develop a higher-level programming language for AED—which, among other things, was used by William Porter and collaborators in 1965–1968 to develop DISCOURSE, one of the earliest urban analysis and simulation software systems (Porter 1969).

Perhaps more consequentially, the plex creeps into the frame of Ivan Sutherland’s “Sketchpad”—the pioneering interactive graphics system which innovatively featured an early form of object orientation for graphical entities (Sutherland 1963). While Sutherland’s acknowledgment of Ross’s contribution is carefully restricted to advice on the “n-elements”—a method of structuring storage memory in the TX-2 computer’s registers to allow for efficient data retrieval—Ross’s diary entries describe a mentoring relationship in which Sutherland is portrayed as an eager disciple. “Ivan said he was very enthusiastic about this [a discussion on pointers] and will keep checking back as I am the only one that seems to be able to give him good concrete discussions and a little argument pro and con on various ways of doing things” (Ross 1961: 4). In later comments to Sutherland’s dissertation draft in the spring of 1963, Ross uses a less mellow tone to express concern that his contribution to Sketchpad’s were not properly acknowledged:

This direct [and enthusiastic] acknowledgment of the [priority and] correctness of my ideas, viewpoints, and techniques make the more recent minimization of my profound influence on Sutherland and the Sketchpad system very painful and difficult to reconcile [or understand] (Ross 1963: 3).³

³ Bracketed statements are hand-written annotations on the typed diary entry.

One of many points of concern for Ross was that the implementation of “master” and “instance” in Sutherland’s thesis resembled private discussions between the two about “generics” and “specifics:”

The importance of the distinction between generics and specifics was emphasized from the very beginning [of the advising relationship]. We discussed these things extensively in the project prior to the Point-Line Diagram study, in our initial investigations using the LISP system, including the generation of new instances and copying of normal case information from the general element. These things were all known to Ivan (Ross 1963: 4).

Of course, the politics of technological authorship at the MIT CAD Project are probably less clear than what Ross—or Sutherland—would have us believe. Complicating Ross’s numerous claims to the paternity of computer-aided design, Timothy E. Johnson—MIT CAD Project member, Sutherland contemporary, and developer of Sketchpad III—told me simply in an interview “we never used any of the Ross stuff” (Johnson 2014). By the 1980s Ross had expanded the theoretical reach of the plex abstraction to help assuage the anxieties of MIT students—it is not entirely unreasonable to infer that his 1986 IAP course on “plex philosophy” helped participants cope with the heavy workloads and deep anxieties of life at MIT.

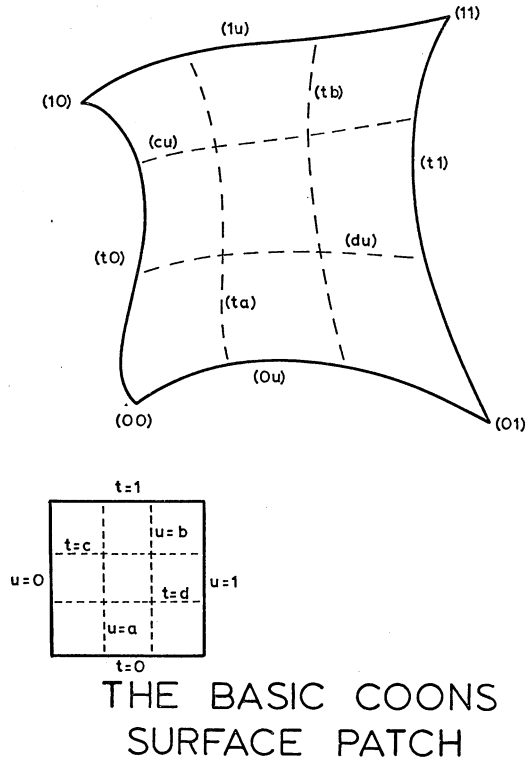
Here lies the powerful ambivalence of the plex; while used tactically in pursuit of a higher language of abstraction, the plex construct was also a way for Ross to index his expertise and his precocious envisioning of hierarchical data structures and object-oriented programming; it was, perhaps, a device to stake claims on concepts and methods of computer-aided modeling and design circulating across MIT’s postwar research ecosystem. It was also, crucially, a managerial device that helped discipline the efforts of cohorts of engineers under his command: a banner, so to speak.

How a plex *represents* is of special relevance to our analysis. The plex delineates an approach to modeling that eschews the description of an artifact’s physical or geometric attributes as its chief purpose. It emphasizes instead the architecture of the relationships between the different elements—physical or not—that comprise an artifact. In doing so, the plex elicits an understanding of computational design premised on the structured nature of digitally encoded information. This understanding foreshadows and continues to underpin ongoing sensibilities in architectural modeling—in particular, those linked to building information modeling, where form is only one of many descriptive layers in a complex arrangement of information.

The Patch

The “Coons patch” is an early mathematical technique for interpolating the position of points on a three-dimensional curved surface defined by a set of parametrically-defined curves. One of its origin stories renders its author, the young self-taught mathematician Steven A. Coons, “pushing a broom” as an employee at Chance Vought Aircraft in the 1940s, coming up with a technique to more efficiently calculate the geometry of aircraft parts for manufacturing (Cardoso Llach 2015:

Fig. 3 The Basic coons surface patch (Forrest 1968: Fig 13.1).
Courtesy of Robin Forrest



54). Later, as faculty in the department of mechanical engineering at MIT, Coons described the technique in greater detail in a publication that became known as the “little red book” (Coons 1967), which circulated widely across engineering and manufacturing circles on both sides of the Atlantic. Before French engineer Pierre Bezier’s work at Renault yielded numerically stable methods for describing and manipulating three-dimensional surfaces,⁴ Coons’s technique first offered a path towards the computational modeling and manufacturing of these shapes (Fig. 3).

The patch is the most visible accomplishment of Coons career. This is reflected in the fact that the most prestigious award for computer graphics research is named after him. However, as I argue elsewhere (Cardoso Llach 2015), his precocious and tireless advocacy of design as a cybernetic, interactive dialogue between humans and machines makes him one of the twentieth century’s most influential design theorists.

Coons had a complex disciplinary identity. He was a self-taught mathematician and designer, a charismatic storyteller, and an occasional photographer and artist. As a publicly-engaged academic and co-leader, with Douglas Ross, of the MIT CAD project he had a platform from which to promote design as a form of

⁴ For a discussion of Bezier’s technique see, for example (Díaz Moreno 2020).

human-machine collaboration. He inspired a generation of researchers whose work helped re-shape contemporary understandings of design, geometry, graphics, and interaction. His students included Ivan Sutherland, Nicholas Negroponte, Lawrence Roberts, Richard Riesenfeld, and Robin Forrest, among many others. His patches circulated widely in industry reports, technical papers, and films. They were key to the configuration of an international network of researchers on both sides of the Atlantic around questions of computation, graphics, and design.

This is illustrated by the creation of a research laboratory in the UK modeled on the MIT CAD Group. Key to this effort was Charles Lang, a graduate student involved in the graphics hardware of the IBM 7094 computer at the MIT CAD Project returned to England to head the Cambridge CAD Group in 1965 by request of Mathematical Laboratory director Maurice Wilkes. This coincided with the decommissioning of the first EDSAC computer and with the activation of the Titan computer and the PDP-7 display. As Wilkes worked to emulate the Timeshared machine he had seen at MIT, Lang modeled the new group after the MIT CAD Project. This included an effort to bring the laboratory closer to industry through a program of industry liaisons, which brought engineers from Ford and Boeing among others to Cambridge. Under Lang's direction, the Cambridge CAD Group produced important innovations on data structures for three-dimensional geometric models, computational geometry, including an early library for graphics called GINO (a predecessor to GL) and "B-Reps," a data structure that continues to undergird most commercial solid modeling packages (Braid 1974). The group, which also comprised Robin Forrest—a mechanical engineer who also visited MIT and would carry Coons's legacy forward by formalizing computational geometry as an academic field—Philip Tabor and Ian Braid, also invented an early desktop-sized numerically controlled machine and a laser display. Unlike its MIT predecessor, the CAD Group at Cambridge was not funded by the military, but by a civil body: the British Science Research Council. It was active between 1965 and 1976.⁵

In addition, Coons's patches illustrate how early research into computer graphics rested on an alignment of industrial, academic, and governmental supports. Once programmed and visualized in the phosphorescent screens of the CRT displays, patches were animated, photographed, and recorded in film, and these images and recordings were disseminated across multiple locations in government and industry. For example, as a consultant for Ford Motors and other companies and universities during the 1960s, Coons worked with engineers and designers in industry to disseminate the technique. Later technical innovations included the development in the 1980s of non-uniform rational Bezier surfaces (NURBS) (Nasri 2003). Patches helped promote the computer as a universal modeling machine with a virtually limitless applicability to a variety of fields, including airplane design for defense,

⁵ Further illustrating the transatlantic character of this research network, the work of a young talented researcher at the Cambridge CAD Group, Ian Braid, had a formative impact on the also young Charles E. Eastman, a US architect and computer scientist working at Carnegie Mellon University, who in the following decades would do much to theorize, develop, and popularize the concept of BIM in architecture.

industrial design for car manufacturing, and scientific visualization for research. As importantly, these efforts played an important role in the broader project to civilianize computer technologies by incorporating them into non-military contexts.

Patches were thus visual artifacts, modeling techniques, and emblems for a distinct way of thinking about designing and building. They codified an understanding of computational design founded on the geometric plasticity of computational descriptions, and on the possibility of physically realizing these descriptions via numerically-controlled manufacturing methods.

Discussion

This article has presented two early modeling techniques, the Coons patch and Ross's plex, as historical artifacts with important technical, ideological, and social effects on present modeling practices in architecture and design. It showed how they enact two distinct kinds of plasticity—one formal and another informational—which continue to underpin contemporary sensibilities in architectural design. In addition, they illustrate the productive ambivalence of computational artifacts as technical objects while also revealing them as vehicles of theoretical understandings of design, and as instruments of persuasion and disciplinary demarcation. In closing, let us further discuss these two vectors.

Two Distinct Kinds of Plasticity

If plexes encompassed a view of design based on structured, hierarchical decomposition of artifacts into informational units, and on the abstraction of design parameters into quantifiable attributes and categories, Coons patches inscribed a view of design based on geometric, material, and manufacturing concerns. Their plasticity was geometric and sought to resemble the material affordances of aluminum sheets or metal rods used to build airplane wings, ship hulls, and other industrial artifacts. In contrast, plexes' plasticity was not formal or geometric, but informational. Their descriptive capacity had less to do with its encoding of material or geometric parameters and more with its indexing of potentially multiple types of data in potentially complex networks of attribution and dependency. This conceptual distance, or contrast, was very clear to the members of the MIT CAD Project, and was at the root of a broader debate about the role of computers in design within this group (Cardoso Llach 2015: 60).

On the one hand the plex's privileging of *information* as a modeling material foreshadows contemporary approaches to building information modeling (BIM) in architecture, where information about materials, costs, structural calculations, and logistics are, at least in theory, intertwined with geometric information. On the other, patches privileging of *form* as their key modeling material foreshadows contemporary approaches to geometric modeling in architecture where shapes take precedence over data. These distinct types of plasticity—formal and informational—first exemplified by plexes and patches—can be seen as

foundational to the contemporary landscape of architectural design technologies, and to the different architectural and design identities it has engendered.

The travel of these CAD techniques across disciplines and industries tells us something important about twenty-first-century architectural sensibilities, and their extra-disciplinary origins. First, the techniques for surface representation and manufacturing developed by Coons and his students enabled stylistic innovations prevalent in architecture during the first two decades of the twenty-first century sometimes referred to as “parametricism.” First adopted by car and airplane manufacturers such as General Motors, Lockheed and Ford, these techniques entered the world of architecture as a result of the advocacy of Coons and his students and, crucially, through software. On the one hand, Coons was a remarkably active and effective public speaker and taught two architecture courses at MIT. On the other hand, software systems widely used in architecture, such as McNeel’s *Rhinoceros*, have origins in mathematical engines linked to aircraft industry implementations that included Coons’s technique (Cardoso Llach 2015: 93). From here it is worth pondering what an architectural history of “parametric” architecture would look like if it took the epistemic legacies of the style in postwar era computing research and modeling techniques seriously into account.

Second, in imagining computational representations as informational artifacts, Ross helped define the technical and discursive skeleton of now-dominant practices of building design and construction in architecture. By devising, through the plex, a way of thinking about information modeling, Ross not only foreshadowed object orientation but hinted at an epistemology of computational representation separate from form. Here, architectures of information take precedence over and subsume geometric or formal attributes. As a result, computational modeling becomes conceptually intertwined with resource management. In the field of architecture, this sensibility is most clearly expressed by the paradigm of BIM, which increasingly structures architects’ professional practice. From here it is worth pondering what other overlaps and tensions we might productively identify between architectural and software theories.

Plexes and Patches: Ambivalent Artifacts of CAD

As we have seen, in these distinct ways, plexes and patches illustrate the ambivalence of computational artifacts. We might think of them as *things* that do *things*. Both modeling techniques served multiple purposes and performed multiple functions, from the operative to the ideological. They were important for the formation of social groups and disciplinary identities, drew links across government, industry, and academia, and foreshadowed a contemporary landscape of modeling technologies. This reminds us that when dealing with abstractions it’s worth asking what is *it* an abstraction *of* and who is served by the distance this move imagines between things, materials, labors on the one hand, and notations, symbols, and ideas, on the other.

References

- Agre, Philip. 1997. Toward a Critical Technical Practice: Lessons Learned Ni Trying to Reform AI. In *Social Science, Technical Systems and Cooperative Work*, eds. Geoffrey Bowker, Susan Leigh Star, Les Gasser, William Turner, 131–57. New Jersey, London: Lawrence Erlbaum Associates.
- Braid, Ian C. 1974. *Designing with Volumes*. 2nd rev. ed. Cambridge: Cantab Press.
- Cardoso Llach, Daniel. *Builders of the Vision: Software and the Imagination of Design*. London, New York: Routledge, 2015.
- Díaz Moreno, Fernando. “Hand Drawing in the Definition of the First Digital Curves.” *Nexus Network Journal* 22, no. 3 (September 1, 2020): 755–75. <https://doi.org/10.1007/s00004-019-00472-1>.
- Johnson, Timothy E. 2014. Interview. By Daniel Cardoso Llach, July 7, 2014.
- Law, John. 2002. *Aircraft Stories: Decentering the Object in Technoscience*. Illustrated edition. Durham, NC: Duke University Press.
- Nasri, Ahmad H. 2003. NURBS from the Pioneers. *Computer-Aided Design* 35 (1): 121–22. [https://doi.org/10.1016/S0010-4485\(02\)00092-1](https://doi.org/10.1016/S0010-4485(02)00092-1).
- Porter, William Lyman. 1969. The Development of DISCOURSE: A Language for Computer Assisted City Design. Ph.D. Thesis, MIT. <http://dspace.mit.edu/handle/1721.1/39037>.
- Ross, Douglas Taylor. 1959. Letter from D. T. Ross to G. Pascoe, January 22, 1959. AC 151, Series II, Box 36, Project 683, Correspondence AIA, 1959. MIT Archives.
- Ross, Douglas Taylor. 1960. *Computer-Aided Design: A Statement of Objectives*. Technical Memorandum 8436-TM-4. Cambridge: Electronic Systems Laboratory, Department of Electrical Engineering, MIT. <http://images.designworldonline.com.s3.amazonaws.com/CADhistory/8436-TM-4.pdf>.
- Ross, Douglas Taylor. 1961. Daily Resumes Vol # 2, Dec 8–18, 1961. MC 4141, Douglas Ross Papers, Box 92, MIT Archives.
- Ross, Douglas Taylor. 1963. Comments on Ph.D. thesis of Ivan Sutherland. Daily Resumes Vol # 2, Dec 8–18, 1961. MC 4141, Douglas Ross Papers, Box 92, MIT Archives. MIT Archives.
- Ross, Douglas Taylor, and Jorge E. Rodriguez. “Theoretical Foundations for the Computer-Aided Design System.” In *Proceedings of the May 21–23, 1963, Spring Joint Computer Conference*, 305–22. New York, NY, USA: ACM, 1963. <https://doi.org/10.1145/1461551.1461589>.
- Ross, Douglas Taylor, and John Erwin Ward. 1968. *Investigations in Computer-Aided Design for Numerically Controlled Production*. Final Technical Report ESL-FR 351. Cambridge, Mass: Electronic Systems Laboratory, Electrical Engineering Dept., Massachusetts Institute of Technology. <http://dspace.mit.edu/bitstream/handle/1721.1/755/FR-0351-19563962.pdf>.
- Sutherland, Ivan Edward. 1963. Sketchpad, a Man-Machine Graphical Communication System. Cambridge: Massachusetts Institute of Technology. <https://dspace.mit.edu/handle/1721.1/14979>.
- Forrest, Robin. “Curves and Surfaces for Computer-Aided Design.” PhD Thesis, University of Cambridge, 1968.

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Daniel Cardoso Llach is Associate Professor of Architecture at Carnegie Mellon University, where he chairs the Master of Science in Computational Design and co-directs the Computational Design Laboratory. His work explores issues ranging from social and cultural aspects of automation in design and construction, the politics of representation and participation in software, and new methods to understand design as a socio-technical phenomenon. He is the author of publications, exhibitions, and artifacts—including the book *Builders of the Vision: Software and the Imagination of Design* (Routledge 2015)—and his research has been supported by the Graham Foundation and Canada’s Social Science and Humanities Research Council, among others. Daniel holds a PhD and a MS (with honors) from MIT and a professional degree in architecture from Universidad de los Andes, Bogotá. He has also been a research fellow at Leuphana (MECS), Germany, and a visiting scholar at the University of Cambridge, UK.